NPS-CS-09-004

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DNS and Multilevel Secure Networks: Architectures and Recommendations

by

Paul C. Clark
Timothy E. Levin
Cynthia E. Irvine
David J. Shifflett

February 2009

This material is based upon work supported by the National Reconnaissance Office (NRO) and the Office of Naval Research (ONR). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of that agency.

This report was prepared by:

Paul C. Clark Research Associate	Cynthia E. Irvine Professor
Timothy E. Levin Research Associate Professor	David J. Shifflett Research Associate

Reviewed by:	Released by:
Peter J. Denning, Chair Department of Computer Science	Karl Van Bibber Vice President and Dean of Research

REPORT DOCUMENTATION PAGE			Form approved OMB No 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** February 12, 2009	**3. REPORT TYPE AND DATES COVERED** Research; February 2006 - February 2009	
4. TITLE AND SUBTITLE DNS and Multilevel Secure Networks: Architectures and Recommendations			**5. FUNDING** J448114
6. AUTHOR(S) Paul Clark, Timothy E. Levin, Cynthia E. Irvine, and David J. Shifflett			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Center for Information Systems Security Studies and Research (CISR) 1411 Cunningham Road, Monterey, CA 93943			**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-CS-09-004
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Reconnaissance Office (NRO) and the Office of Naval Research (ONR)			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** Not applicable
11. SUPPLEMENTARY NOTES This material is based upon work supported in part by the National Reconnaissance Office (NRO) and the Office of Naval Research (ONR). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**
13. ABSTRACT (Maximum 200 words.) The Domain Name System (DNS) protocol was introduced to solve a naming problem in TCP/IP networking, namely, to provide a translation service of system names to network addresses (i.e., Internet Protocol (IP) addresses). The protocol was not developed with a requirement to support multilevel secure (MLS) networks. However, the Department of Defense (DoD) vision for the Global Information Grid (GIG) entails support for multilevel networks. In the future, DNS installations must securely deal with multilevel issues. This paper describes specific design recommendations for providing MLS DNS in the context of the GIG Vision, and the Monterey Security Architecture (MYSEA) Testbed. It also describes several other potential MLS DNS architectures along with their advantages and disadvantages.			
14. SUBJECT TERMS Multilevel Secure, Domain Name System			**15. NUMBER OF PAGES** 46 **16. PRICE CODE**
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** Unclassified

NSN 7540-01-280-5800

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18

This page left intentionally blank

NAVAL POSTGRADUATE SCHOOL
CENTER FOR INFORMATION SYSTEMS SECURITY STUDIES AND RESEARCH

CENTER OF ACADEMIC EXCELLENCE

Technical Report NPS-CS-09-004

DNS and Multilevel Secure Networks
Architectures and Recommendations

Paul C. Clark, Timothy E. Levin, Cynthia E. Irvine, David J. Shifflett

February 12, 2009

ACKNOWLEDGEMENTS

This material is based upon work supported in part by the National Reconnaissance Office (NRO) and the Office of Naval Research (ONR). Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

Author Affiliation:

Center for Information Systems Security Studies and Research
Computer Science Department
Naval Postgraduate School
Monterey, California 93943

Table of Contents

List of Figures

List of Tables

Abstract

The Domain Name System (DNS) protocol was introduced to solve a naming problem in TCP/IP networking, namely, to provide a translation service of system names to network addresses (i.e., Internet Protocol (IP) addresses). The protocol was not developed with a requirement to support multilevel secure (MLS) networks. However, the Department of Defense (DoD) vision for the Global Information Grid (GIG) entails support for multilevel networks. In the future, DNS installations must securely deal with multilevel issues. This paper describes specific design recommendations for providing MLS DNS in the context of the GIG Vision, and the Monterey Security Architecture (MYSEA) Testbed. It also describes several other potential MLS DNS architectures along with their advantages and disadvantages.

1 Introduction

The purpose of this section is to provide the reader with enough background information to understand the terminology used and the principles discussed in the remainder of the report. Topics covered are those relevant to naming schemes for the Internet and security concepts relevant to the subsequent analysis and recommendations.

1.1 Name and Address Translation

All computers that participate in a TCP/IP network, such as the Internet, must be assigned an IP network address, which is minimally a 32-bit number [1]. All traffic on the network layer is sent and routed based on IP addresses. A given network node observes all packets that traverse its network connection and processes only the ones whose destination address match its own address. Network nodes that provide a service usually have names associated with them too, because people can remember names more easily than long numbers. For a client computer to request a service on behalf of the user, it must be able to translate the **named** source provided by the user to the **numbered** IP address of the server. The Domain Name System (DNS) protocol was developed to transparently provide this translation. When a user on a client references another system by name, the client queries the configured DNS server for the IP address of the given name. Upon receipt of the corresponding IP address, the client can then send a transmission to the requested destination.

A computer implementing the DNS protocol provides a name-to-address mapping service (and vice versa), and is referred to as a DNS server. A DNS server maintains a database of names and matching IP addresses. When a client computer is configured for a network, the IP address of a local DNS server must be provided if the client is ever expected to dynamically map names to IP addresses.

The high-level concept of the DNS protocol is fairly straightforward, but the details, configuration and other added functionality make it much more complex. For example, there is **no** single DNS server where all the names and addresses of all the computers on the Internet are maintained. The DNS protocol specifies a distributed database made up of many DNS servers.

1.2 Distributed DNS Databases

Each computer name on the Internet is part of what is called the “domain name space”. Distribution of the DNS database occurs by breaking up the domain name space into smaller domains, which in turn can be broken up further into other sub-domains. The top-level domains are familiar: .com, .edu, .net, and so forth. When the XYZ University connects to the Internet, its DNS domain might be xyz.edu. If the university staff would like to manage their own DNS server, the authority to do so is delegated to them by the .edu authority, and an entry in the top-level .edu domain would be made, mapping the xyz.edu name to the XYZ DNS server IP address. The university may manage all mappings on their campus computers, or they could in turn delegate to another level, such as to the Computer Science (CS) department, creating a third-level domain, cs.xyz.edu. The xyz.edu DNS server would no longer contain mappings for the CS computers; instead it would contain an entry that maps the CS DNS server name to that server’s IP address.

When a client in the XYZ CS department needs to translate a name to another computer in the CS department, it queries its local CS DNS server, which returns the translation. If a CS client needs to translate a name to a computer in the Physics department, it contacts the CS DNS server, and then the CS DNS server queries its parent DNS servers. Then, if the Physics department also had its own DNS server, the CS DNS Server would then query the Physics DNS server, which would then return the translation. In other words, given the fully qualified name of a computer (i.e., its full name with respect to the domain name space, such as client1.physics.xyz.edu), there is enough information to walk up and down the distributed DNS tree to find the server from which the answer to the query can be obtained.

When it is necessary to update DNS data for a domain (e.g., when a host is added or removed), the change is made in the appropriate *authoritative* DNS server by the delegated organization such that queries for the IP address can be properly serviced. This new information may also be replicated automatically to *slave* DNS servers to provide redundancy and performance benefits.

DNS servers also support the determination of the IP address of a domain’s e-mail server. If an e-mail is addressed to someone@acme.com, it is unlikely that Acme’s mail server is on the acme.com host. Instead of trying to connect to the e-mail port of the acme.com host, the forwarding system requests what is called an MX record from the DNS server that has the mapping of Acme’s name. The MX record contains the IP address of Acme’s mail server.

1.3 Iterative and Recursive DNS Requests

The client program or library that understands the DNS protocol and makes DNS requests is known as a *resolver*. Most resolvers only have enough intelligence to ask its configured DNS server to return the IP address of a given name. This is known as a *recursive* request, because the resolver needs the local DNS server to do whatever it takes to resolve the name, even if it means asking many other DNS servers for help. If the local DNS server knows the answer, then it returns it to the resolver. If it does not know the answer, then it has enough information to query other DNS servers one at a time until it is resolved. These individual one-at-a-time requests are known as *iterative* requests.

The importance of this distinction between recursive and iterative requests is that for the former a client will make one request to a server and wait for an answer, while the local DNS server often communicates with other DNS servers to get the answer. In other words, both client-to-server and server-to-server communications must be supported.

1.4 DNS Implementations

BIND (Berkeley Internet Name Domain) is the most commonly used DNS software on the Internet, and has been available since 1987. [2][3] It is considered the reference implementation of the DNS protocols. [4] BIND 9 is the current major release, and is large and complex open source software with almost 400,000 lines of code[1] across 1100+ files. BIND 9 constitutes a major rewrite of the previous release, and includes new features, such as support for the larger IP version 6 addresses [5] and additional security settings.

The BIND distribution provides three separate components: 1) the DNS daemon that performs the DNS service, 2) a DNS library for application development, and 3) an array of tests to verify the proper operation of the software. The code dedicated to the daemon itself is approximately 72,000 lines of code.[2] Because of its "reference implementation" reputation, BIND implements all specified DNS functionality. In addition to the complexity of the software, the syntax for the DNS daemon configuration file has a reputation for being complex. There are, of course, other open source DNS implementations [4], some of which claim to be BIND replacements, but with less functionality, which may or may not be desirable.

In addition to open source DNS implementations, there are commercial products available. With respect to security, perhaps most notable of the commercial products are those from Secure64, which claims that their products are "designed from the ground up for availability, security, and performance." [6] However, their products have not been evaluated against an accepted security standard (such as the Common Criteria [7]) to verify their claims.

1.5 Mandatory Access Control (MAC)

Access control policies can be grouped into one of two categories: 1) Discretionary Access Control (DAC) and 2) Mandatory Access Control (MAC). In addition, supporting policies provide for accountability and other security requirements. DAC policies provide opportunities to modify access control settings, such as who can access an object or the name of the object owner, with implementations that present a run-time interface to make those modifications. DAC implementations can be found in various

[1] This figure was determined using BIND version 9.4.0a6. It was a "raw" count which included comments and white space over all .c and .h files.
[2] This is a result of a "raw" count of the source and header files in the `bind-9.4.0a6/bin` directory hierarchy of the source tree.

forms on most modern operating systems, such as recent versions of Microsoft Windows, and all varieties of Unix.

MAC policies, on the other hand, enforce static policies and provide no run-time interface for policy modifications, such as the sensitivity level of a subject or object, or for changing the rules of access. MAC policies have application in government and commercial environments. From a government point of view, files can be classified at various levels of confidentiality, such as Secret and Top Secret. Once a file has been marked as Secret, the owner's desire to share the file with others is constrained to those who are authorized (cleared) to see Secret documents, hence the name "mandatory". If computers or networks support a MAC policy, they are said to be Multilevel Secure (MLS).

When a computer enforces a MAC policy, special problems must be addressed at the policy and data labeling level. For example, the policy must clearly resolve the following: if a file is classified as Secret, but a user trying to access it does not have the necessary clearance: 1) Should the user be allowed to even see that the file exists?, or 2) should the user get a "you do not have permission" error, or 3) should he get a "this file does not exist" error? If the existence of the file should be hidden from those without adequate authorization, it would require the system to "lie" to such people by saying it does not exist. Conversely, if the existence of the file is to be exposed to unauthorized users, then modulation of the file's existence or metadata can result in covert channels through which information can be leaked from a high sensitivity level to a low one [8]. Such MAC policy questions boil down to the following: 1) Is metadata (such as file names and the presence of files) to be protected by the MAC policy? If so, 2) How is static metadata to be labeled? One approach is that all static metadata must be labeled at the same level as the data it describes. Another approach, called "compatibility", is that static metadata must have a confidentiality label that is dominated by the data's confidentiality label [9], i.e., the metadata must be labeled at or below the level of the data it describes. Dynamic metadata must be managed in a way that does not introduce covert channels into the system [10].

The Bell and LaPadula model was developed to formally show the soundness of a MAC confidentiality policy, as described above, which is capable of being enforced by a computer. [9] The model, among other things, stipulates properties that must hold true, which are translated into the two rules given below:

- <u>Read Down</u>
  A subject can only read objects at or below its current sensitivity level, otherwise known as the subject's *access class*.
- <u>Write Up</u>
  A subject can only modify objects at or above its access class.

The latter rule is necessary to prevent subjects at a higher access class from writing information from objects at a high access class into objects at a lower access class.

Supporting policies are those that are needed for the proper operation and oversight of DAC and MAC policies. For example, DAC requires an Identification and

Authentication (I&A) policy to be in force because they depend on knowing the identity of the user to make access control decisions. MAC requires I&A so that a session level can be set that is commensurate with the user's clearance or authorizations.

1.6 Session Level

When a user is added to an MLS system, the administrator must designate the user's clearance, which allows the system to know the confidentiality upper bound for that user. However, when considering the rules of the Bell and LaPadula model, and how they affect an actual implementation, it is not advisable to have subjects running with the absolute clearance of a user at all times. Sometimes a user with a Secret clearance may need to write Unclassified information into an Unclassified file, but any software used on behalf of the user may not be trusted to not write Secret information into the Unclassified file. Therefore, to prevent against undesired flows of information from high sensitivity to low sensitivity, subjects are restricted to run at a single level, dominated by the clearance of the user, and the user is required to specify the access class of the subjects running on his behalf. Setting this specified access class is known as setting the *session level*.

For example, if a user wants to write to a Secret object, the user sets the session level to Secret. The user may still read objects at or below the session level, but can only write to Secret objects, per the Bell and LaPadula rules. If the user needs to write to Unclassified objects, the Secret session must be exited, and a new session must be established at the Unclassified level. The user is then unable to read or write to objects above the Unclassified level, but can read and write objects at the Unclassified level.

1.7 Confidentiality Levels versus Confidentiality Categories

Most people are familiar with confidentiality *levels*, such as Secret and Top Secret. There are a limited number of them. These levels are hierarchical in nature, meaning that given two levels, either one of them will represent a greater confidentiality than the other, or they will be equal. There is, however, a way of providing more granularity and control of access by introducing, in addition to levels, confidentiality *categories*.

A confidentiality category is often a marking that provides a greater description of what the data relates to, or what kinds of people should see the data. For example, a file could have an access class that consists of the Secret level and the Nuclear category, perhaps indicating that the file contains information that is related to a nuclear subject, or should only be viewed by those engaged in nuclear research, depending on how an agency defines its categories.

An object can be assigned only one confidentiality level, but it can have any number of categories, including none at all. Categories are not hierarchical, thus two different categories cannot be compared to see which has "more" confidentiality. However, categories are used to enforce mandatory "need to know" policies. Sets of categories may be assigned to both subjects and objects. If an object has a given category, then the categories in the subject's access class must be a superset of the categories assigned to the object for the subject to read the object (as well as the usual access check on the confidentiality level). Because there are a large number of categories within the DoD and the intelligence community, the possible number of unique access classes is large, considering all the permutations of levels and categories.

1.8 DoD System-High Networks, Domain Names and IP Addresses

The Department of Defense (DoD) operates several system-high networks, such as the NIPRNET at the sensitive-but-unclassified level, the SIPRNET at the Secret level, and JWICS at the Top Secret level. For security reasons, these networks are designed to have limited connectivity, if any. For example, devices exist between the NIPRNET and SIPRNET that permit limited information flow so it can only go from low to high, while JWICS is intended to have no physical connections to lower-level networks.

Even though these networks do not really offer services to each other, and have little or no physical connections, the DoD policy assigns IP addresses and domain names in such a way that they remain unique across all networks [11]. There appear to be two benefits for this policy: 1) if a high-level system accidentally connects to a low-level network (or vice versa) there is a smaller chance that data will leak inappropriately; and 2) if all the networks really do collapse into one integrated network in the future, the task of re-addressing a large number of systems is avoided.

Dividing a DNS domain into sub-domains according to access class provides the obvious benefit of helping users see the access class of a host address, and it also makes it easy to send an iterative DNS request to the correct server. For example, the DoD uses the ".mil" DNS domain as its root, the ".smil.mil" as a Secret DNS sub-domain, and perhaps ".ts.mil" for a TS DNS sub-domain.

1.9 MYSEA Testbed

The Monterey Security Architecture (MYSEA) Testbed was built to support research on MLS services, clients and networks [12]. It consists of a federation of high assurance MLS *MYSEA Servers* that connect to several single-level networks running at different simulated confidentiality levels, as well as a single MLS network. By changing their session level, users on the MLS network can access the single-level networks without moving to a different client. In return, users on the single-level networks can access services on MYSEA Servers configured to respond to their access class.

A primary objective of the multilevel testbed research project is to demonstrate how U.S. participants can use a single workstation for multilevel access to U.S. and coalition WANS at different classification levels. Currently, the testbed supports experimentation with access to multilevel as well as multiple single level (MSL) networks. It supports commercial office productivity applications in the context of high assurance multilevel security. The Testbed also supports experimentation and development of MLS aware applications in the context of high assurance MLS policy enforcement and dynamic security services, two areas that are critical to the realization of the DoD's vision for assured information sharing [12].

The MYSEA server acts as a nexus for MLS policy enforcement and for communication between the MLS network and the several single-level system-high networks. For users on the MLS network, it provides a variety of MLS services, such as e-mail and web services. This is done by securely binding a client's IP address to the user's session level. For example, when a user requests a web page, the MYSEA Server spawns a web daemon with the user's credentials and session level. The daemon is restricted by the policies enforced by the underlying server.

The MYSEA Testbed is prototyping a multiple single-level-at-a-time (SLAT) client that can interact with the MYSEA server in a secure manner. An MSL client can operate at a particular session level at one time, but can read information at a lower sensitivity level and perhaps write up, e.g., send e-mail to a higher sensitivity level.

The testbed is currently limited in its DNS capabilities. Each single-level network can run its own DNS server to service systems on its respective network. While the clients on the MLS network can be configured to use one of the single-level networks for DNS, they cannot access a DNS server that manages all the available access classes without modifying the DNS settings after each change of session level. In addition, there is no multilevel DNS service provided by the MLS Server, for example, so that a client on a Secret network could read DNS information from an Unclass DNS server.

1.10 Global Information Grid

The Global Information Grid (GIG) is a DoD network of networks. The GIG Vision [13] is an attempt to describe the desired functionality of the GIG in the year 2020. In this vision, the system-high networks (e.g., NIPRNET and SIPRNET) are collapsed into one secure MLS network with clients that are either dedicated to a single access class or an MSL client.

1.11 Covert Channels

A covert channel is an unintentional method of communicating data between two or more parties by manipulating return values or timing changes in relation to a sequentially accessed shared resource. Covert channels are of special interest in MLS systems because they can potentially provide a way to bypass the enforced MAC policies. For example, a high-level subject can transfer a bit of information by either filling up a disk, or not filling up a disk, while a low-level subject is either successful creating a new file (or unsuccessful) at the agreed upon time. Systems that enforce MAC policies must be carefully designed to eliminate or seriously impede covert channels.

1.12 Trusted Computing Base

The hardware, firmware and software that are trusted to enforce a system's security polices are known as the *Trusted Computing Base* (TCB). The *security perimeter* is the logical boundary between the TCB and the untrusted parts of the system.

1.13 Trusted Subjects

An MLS system should enforce its policies in the lowest layers of the overall system, such that they are always invoked when a subject tries to get access to an object. There can be designs, however, that require applications to support a MAC policy. Generally, these are in higher layers within the overall system. In such cases, these applications are referred to as *trusted*, because they are trusted to ensure that the intent of the security policy is observed outside the kernel or operating system. Having trusted applications extends the security perimeter to encompass those applications and any parts of the system the trusted applications depend upon. Trusted subjects usually permit something contrary to the enforced policy to occur, but they are trusted to observe the intent of the policy. When dealing with sensitive data, such applications must be worthy of that trust, which only comes from adhering to rigorous development practices that address security concerns. High assurance can be achieved by following requirements derived from well-known standards, such as those described in the Common Criteria (CC) [7]. In general, a well-designed MLS system limits its dependence on trusted subjects, if they are used at all. For example, modifying an open source web server to enforce a MAC policy of some sort would make the web server a trusted subject, but it is not worthy of that trust, and is likely a bad design choice. Just because it is trusted does not mean that it is trustworthy.

2 The Problem

This section provides specific direction and MLS DNS requirements, as well as policy assumptions and design considerations.

2.1 Research Scope and Focus

This research focuses on the confidentiality of DNS data, as it is requested from clients that are operating at different access classes. It does not focus on other important aspects of DNS security, such as integrity and authenticity of name resolutions.

2.2 Policy Assumptions

The security policy, with respect to IP addresses and network device names, is assumed to be the following:

1. While servicing clients at its sensitivity level, a DNS daemon can read lower-level DNS data files and provide the mapping to the requesting client.

   For example, a Secret subject may obtain the IP address of an Unclassified network device if it knows the name of the device. Preventing the actual TCP/UDP connection from high to low is a network policy enforcement issue that is not considered in this report.

2. A lower sensitivity level DNS client cannot resolve higher sensitivity level names, mappings, or other metadata.

   To limit covert channels, the client must get the same response whether the name exists or not.

3. Fully qualified domain names (i.e., the host name concatenated with its DNS domain name) are unique across an MLS enterprise network, as one would expect from the IP specification.

2.3 MLS DNS Requirements

The following list describes the requirements that an MLS DNS design must satisfy.

1. Hosts shall be able to query a configured DNS server to resolve a name to an IP address without leaking information from high to low. This includes hosts that operate at a single access class as well as hosts with MLS capabilities.
2. DNS daemons shall be able to perform iterative queries that pass on enough information to other DNS servers (e.g., the original requestor's access class) to allow the queried DNS server to make appropriate MAC policy decisions.
3. DNS data shall be protected from observation and modification by unauthorized subjects when stored on DNS servers. It is assumed that encryption is used to protect network communications on MLS networks, commensurate with the level(s) of data transmitted.
4. Hosts requesting DNS data that exists at lower sensitivity levels shall be able to obtain it (i.e., perform a read down).
5. Access to DNS data shall be controlled based on the access class of the requesting subject and the access class of the DNS data.
   We desire a DNS service in which individual DNS records may be labeled so that a coherent Enterprise IT security policy is possible. This labeling must be on a record-by-record basis, a file-by-file basis, or a system-by-system basis (e.g., a particular DNS server only has names and IP addresses for one particular access class). No matter which granularity is used, enforcement of the MAC policy must be based on the access class of the requesting subject and the classification of the requested DNS data.
6. The assurance provided in components that store or resolve DNS data shall be commensurate with the confidentiality of the data and the operation of other subjects running on the component.
7. The MLS DNS design shall be able to scale to a large number of access classes, both in terms of DNS data and requesting subjects.

2.4 Some Implementation Considerations

To illustrate some of the MLS DNS issues, Figure 1 shows a *potential* MLS network, where there exist single-level networks attached to an MLS network backbone, as well as a mix of MLS clients and servers. The figure shows the DNS resolution for PC2, that will allow PC2 to communicate with PC1, where both PC1 and PC2 are operating at the Secret level. Because PC2 does not have the destination IP address for PC1 it contacts its local DNS server to perform a search which is recursive from the perspective of PC2 but iterative at the local DNS server (step 1). The local DNS server does not know the complete answer, but the information provided is sufficient to cause it to contact an MLS DNS server (step 2). However, the MLS server cannot complete the answer either, but it does provide to the local DNS server the IP address of a server that should know the IP address for PC1 (step 3). The local DNS server queries the DNS server in the destination domain for the IP address of PC1 (step 4), which it returns (step 5). The local DNS server returns the desired IP address to PC2 (step 6), which then communicates with PC1 (steps 7 and 8), by sending an IP packet with that destination address onto the network.

“Broadcasting” of IP packets on an MLS network raises the question of MLS policy enforcement and covert channels. It is assumed that the payload of messages higher than the level of the network will be suitably protected (encrypted), and that protection against traffic analysis of hosts higher than the level of the network will be provided.

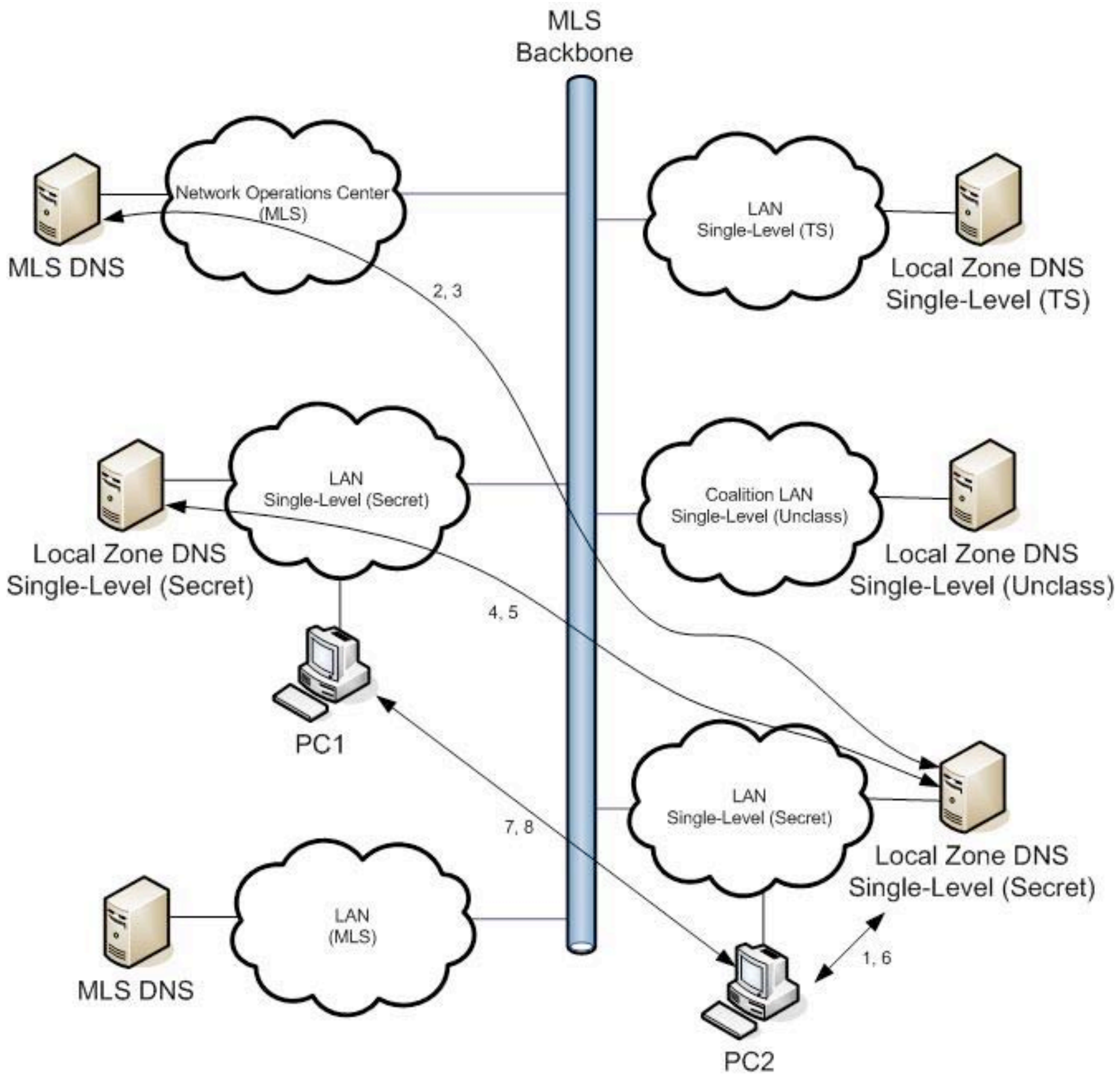
MLS Backbone
Network Operations Center (MLS)
MLS DNS
2, 3
LAN Single-Level (TS)
Local Zone DNS Single-Level (TS)
LAN Single-Level (Secret)
Local Zone DNS Single-Level (Secret)
Coalition LAN Single-Level (Unclass)
Local Zone DNS Single-Level (Unclass)
4, 5
PC1
7, 8
LAN Single-Level (Secret)
Local Zone DNS Single-Level (Secret)
LAN (MLS)
MLS DNS
1, 6
PC2

Figure 1. DNS in an MLS Context

The point of Figure 1 is to show that single-level networks sitting off an MLS backbone may have their own single-level off-the-shelf unmodified DNS servers that may be configured to connect to root DNS servers that operate across many levels. Local networks that have MLS capabilities will need a local MLS solution, which must be able to communicate with single-level DNS servers in other networks, as well as other MLS servers. In addition, the root DNS server(s) will need to be MLS because they are typically responsible for communicating with all DNS servers (at all access classes) within its DNS domain.

Another issue, depending on the design, is that there may be a requirement for the DNS server to know with a high degree of confidence the access class of the subject making the DNS request so that a lower-level subject cannot obtain higher-level DNS data. In such designs a secure and non-spoofable protocol would have to be developed for providing the access class to DNS servers.

3 Recommendations

After considering a number of approaches for enabling DNS in a MLS environment, the following subsections describe the recommendations for DNS in two specific and separate environments: the MYSEA Testbed and the DoD GIG Vision. The MYSEA Testbed and the DoD GIG Vision were briefly described in Sections 1.9 and 1.10 respectively. For comparison with the recommended approaches, Appendix A describes other possible MLS DNS architectures that were considered.

In addition to the policies and requirements discussed in Section 2, a number of factors affect prospects for near term implementation of MLS DNS. The ideal solution would be one that required no modifications to current DNS software or protocols and was highly trustworthy. Thus, a solution that minimizes modification and disruption of current DNS architectures is sought. In addition, issues associated with the system’s operational environment as well as resources available for its construction must be considered.

3.1 MYSEA Testbed

The recommended solution for near-term demonstration on the MYSEA Testbed (described in Section 1.9) uses a custom DNS proxy and an unmodified single level COTS DNS service per level on each DNS platform. Each daemon has its own file of DNS mappings. The proxy is a trusted subject that routes the request to the appropriate DNS daemon and enforces the MAC policy with respect to the access class of the request and the access class of the DNS daemon, as shown in Figure 2. The access class of a DNS request is based on the client IP address. Changes to a client session level will require a corresponding update to the DNS proxy’s table of IP address levels. This is similar in design to the Secure Security Services (SSS) already implemented in the MYSEA Testbed.

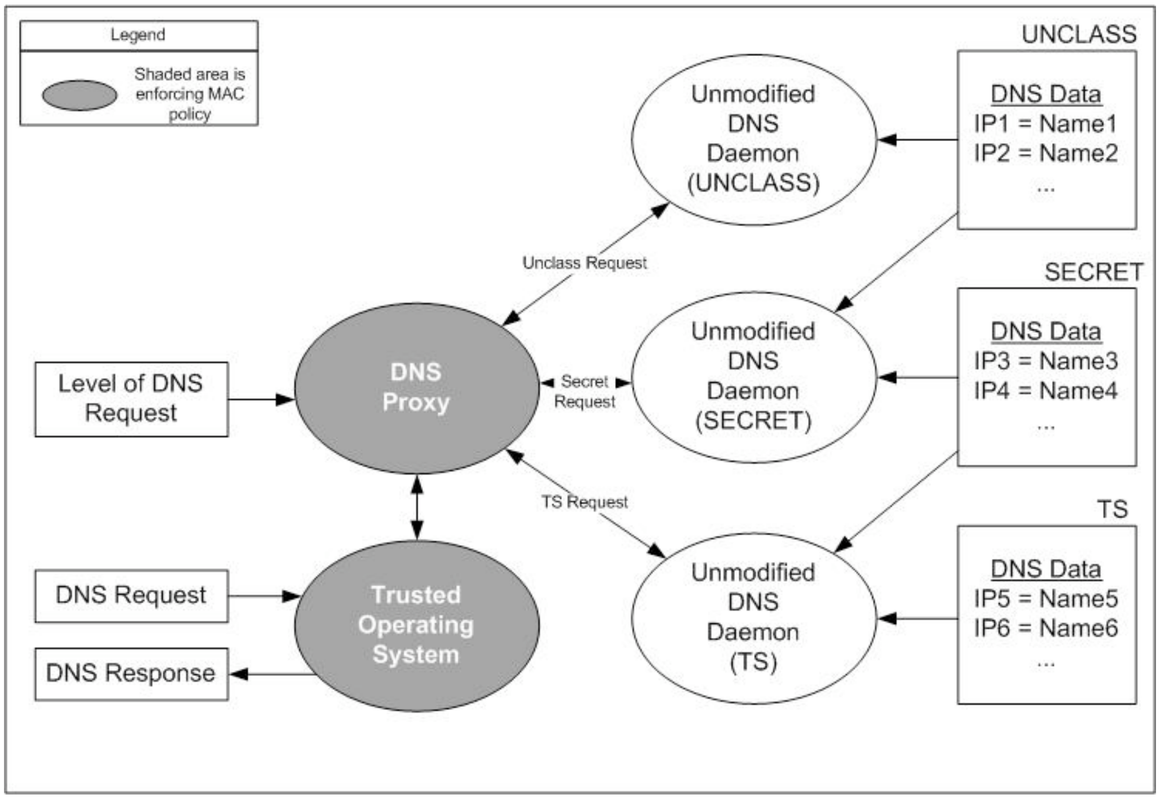

Figure 2. Recommended Approach for MYSEA Testbed

This approach is appealing because MYSEA already has functionality similar to the described proxy running on a high assurance MLS server, and the MYSEA Server already maintains the session level associated with client IP addresses. The DNS proxy could be used to start DNS daemons on the MLS server to answer queries about the MLS domain, or forward DNS queries to the appropriate DNS server on a single-level network for resolution in other zones. The OS ensures that the daemons can only access the appropriate DNS databases, and the multilevel proxy must forward requests to appropriate single level daemons.

With multiple DNS daemons running on one physical platform, this approach will use more resources than a normal single instance of DNS. The concern about scaling up to some large number of supported access classes on the MYSEA Server can be reduced somewhat because the proxy could be modified to only start DNS daemons at access classes that are actually requesting DNS data. For example, if DNS requests are only being made from Unclass and Secret clients, then instances of the DNS daemon will only be started at Unclass and Secret; the DNS daemon could be left running as long as requests continue to arrive for its access class. The proxy could be modified to stop the daemon after a configured amount of time has elapsed with no DNS traffic arriving for that daemon. Or, the DNS daemon could be modified to exit after a specified period of inactivity.

With respect to the question of DNS daemons being able to read lower-level information, there are two possible solutions for this MLS DNS proxy, Single Level Daemon approach. In the first solution the daemons can be configured to read the lower sensitivity DNS data, as long as the names associated with the lower sensitivity level IP addresses are in DNS zones that are distinct from higher sensitivity level zones, e.g., host.mil versus host.smil.mil, otherwise there could potentially be one name that exists at multiple levels that maps to different IP addresses at each of those access classes. Instead of reading multiple DNS mapping files, the second solution for reading lower sensitivity level mappings is to merge all the appropriate DNS data files for a daemon into one file prior to the start of each daemon (i.e., merge only the files that are at or below the level of the starting daemon). The merging could be done intelligently by performing a new merge of the data only if one of the input files has changed. An added benefit to this second solution to reading lower-level information would be the identification of IP addresses and host names that have mistakenly been assigned to more than one access class.

DNS policy decisions are MAC-based, so policy decisions do not require the name of the user making the request; the DNS proxy only needs to know the level of the original DNS request. Therefore, if iterative DNS requests will take place between two or more MLS DNS servers, then a protocol for passing the access class of the request between proxies must be created.

A positive side effect of the MLS DNS proxy, Single Level Daemon approach is that system administrators do not need to be concerned with the access class of DNS servers when configuring DNS clients; all clients on single-level or MLS networks, regardless of access class, are configured with the same MLS DNS Server IP addresses. This removes the possibility of mis-configuring a client with the IP address of a higher-level DNS server.

Unfortunately, this approach to implementing a MLS DNS solution in the MYSEA Testbed extends the security perimeter beyond the OS to include the proxy application, since it helps to enforce the MLS policy. However, this approach might not require modifications to existing DNS software (depending on how the proxy is designed, and how the existing DNS software can work with a proxy), but it does require the development of new software, i.e., the DNS proxy.

3.2 DoD GIG Vision

The DoD GIG Vision has an overall design that is similar to the MYSEA Testbed, namely, it has single level networks at varying access classes that are connected to a common MLS backbone. It is also envisioned that there would be some number of MLS clients and servers. [13] Therefore, one MLS DNS solution would be the implementation of the MYSEA architecture, as described above. If the MYSEA architecture is not suitable for one or more reasons to be determined, this section describes an alternative approach for the DoD GIG Vision.

This approach uses an (to be determined) operating system (OS) to interpret destination IP addresses and route DNS requests to a single-level DNS daemon. There is one DNS daemon per access class, but they are all running on one physical server (or some set of servers, depending on the load), as shown in Figure 3.

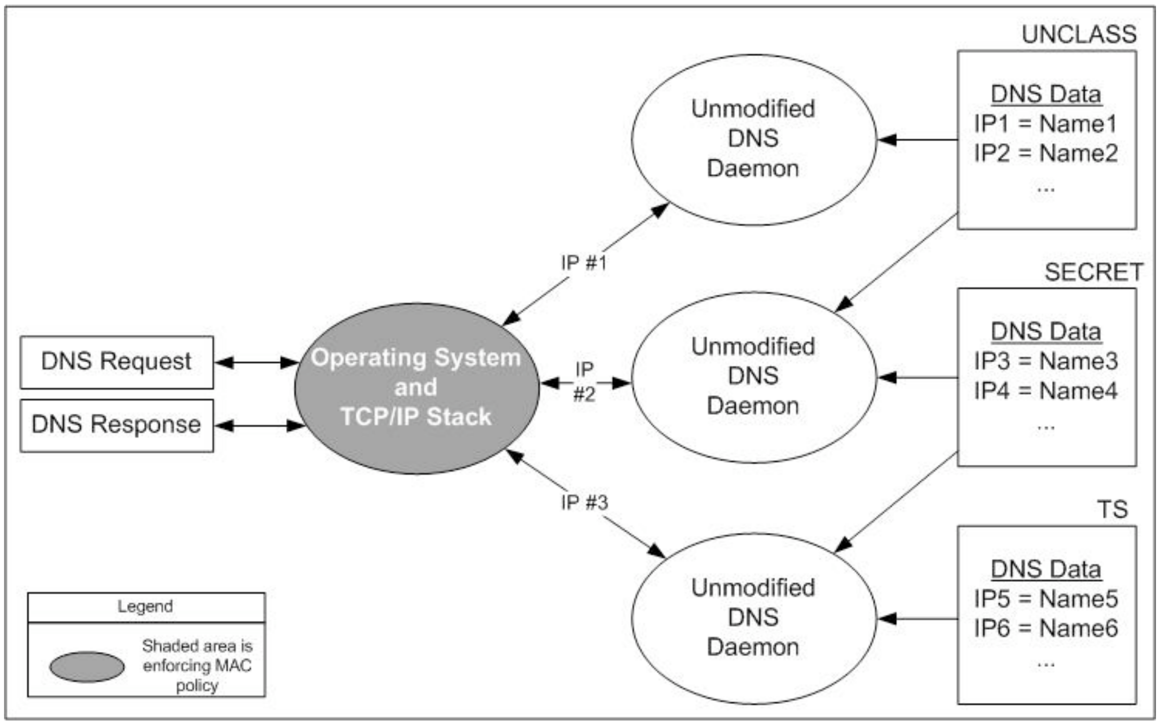

Figure 3. Recommended Approach for DoD GIG Vision

This is similar to the MYSEA Testbed recommendation, described in Section 3.1, but the differences are subtle and important. In this approach, the mechanism for multiplexing the DNS request is the operating system (not an application-level proxy). The OS routing decision to a DNS daemon is based on the **destination** IP address of the requesting packet rather than the access class of the requester (as with the MYSEA approach).

Each DNS daemon would have to be configured to listen to a different IP address, because each is listening to the same TCP/UDP port. The operating system would have to support the ability to assign multiple IP addresses to the same network interface card (NIC), which is a feature that may not be currently available. (Note that having multiple NICs, with one access class per NIC does not scale well, nor is it flexible enough to accommodate dynamic changes to the number of access classes as policy evolves). Each client would be configured with the IP address of the correct DNS server for its access class, so there is no need to dynamically communicate the access class of the request to the DNS daemons, which is very appealing because it eliminates the need to develop new software or modify existing DNS software.

A different view of this approach is to use a high assurance separation kernel [14][15] as the trusted decision maker, which forwards the requests to daemons running in separate partitions. The daemons would return the responses to the decision maker to forward to the requesting client.

As with the MYSEA solution: 1) having one DNS daemon per access class requires more hardware resources than the normal single instance per DNS server, so scaling to a larger number of access classes would require additional servers to handle the load; 2) this approach also requires a high assurance MLS server to ensure that the untrusted DNS daemons servicing lower-level access classes cannot read the higher-level DNS data files; and 3) the tradeoff for not needing new DNS daemon software must be balanced with the cost of buying MLS servers, because high assurance MLS servers tend to be expensive and feature-poor.

Supporting DNS requests from MSL or MLS clients becomes an issue because the destination IP address for the DNS server depends on a user's changing session level. An MLS client can only be supported in this approach if the client OS, or a client proxy like the Trusted Path Extension (TPE) [12], can dynamically change the destination DNS server IP address based on the session level of the requesting subject.

This approach can easily deal with the problem of recursive lookups by DNS daemons. All daemons would be configured to query DNS servers (via a destination IP address) that store the same level of information. Therefore, no changes in DNS protocols are required to support it.

This approach assumes that the GIG Vision network will have some method of identifying and handling inappropriate requests (e.g., a deliberately misconfigured lower sensitivity level client that tries to resolve names with a DNS server that is meant to service higher sensitivity level requests). For example, there could be a cryptographic binding of an IP packet to an access class, which can be verified by a trusted component.

Currently, there is no appropriate high assurance OS available that can support the solution described in this subsection.

4 Previous Work

No previously published research was found that directly addressed the problems of DNS in an MLS environment. There are, however, publications that describe multilevel solutions for related functions. These are summarized below.

4.1 Multilevel Relational Databases

Beginning with the 1982 "Summer Study" [16] much has been researched and written about securely supporting multilevel data in a relational database [17][18][19]. In such a database, the records in a table will likely have various classifications. The results of a database query would therefore be dependent on the access class of the subject making the request; a higher-level subject would see records at more sensitivity levels than a lower-level subject, as expected.

An unexpected quality of a multilevel database is the result of inserting records into the database, similar to the problem of handling metadata in an MLS file system (see Section 1.5). If a subject at a lower level tries to insert a new record, but the primary key of the new record already exists at a higher level, what should the database do? If the record is accepted, then two records exist with the same primary key – something that goes against the "rules" of relational databases. On the other hand, if the record is not accepted, as one would expect, the lower level subject is able to infer knowledge that exists at a higher level. This could be used as a covert channel for information leakage. In this situation the SeaView model [19] accepts the record with the duplicate key by considering the primary key to be a combination of the defined field and the classification of the inserted data. This existence of two records with the same *traditional* primary key is known as *polyinstantiation*. However, due to the current understanding of how the DoD networks are managed, names and IP addresses are unique across all classified networks, so there should be no polyinstantiation issues to deal with in the DoD multilevel DNS architecture. There may indeed be system administrators at varying levels of clearance inserting records into the DNS data files for their local name space, but the operational management of the names and addresses (i.e., the distribution of IP addresses by a central high-level office) should prevent a situation where two or more different answers to a query are possible, depending on the subject's access class.

4.2 Multilevel DoD Directory Services

There was some prior research performed to design a multilevel directory service for the DoD. "Directory service" in this context refers to an X.500-style "white pages" and "yellow pages" service where information about individuals or other entities could be stored and publicly queried. A prototype directory service was implemented on a prototype implementation of the SeaView model. [20]

The X.500 architecture consists of a series of distributed and "hierarchically organized" servers (known as Directory System Agents, or DSAs) that can either answer a query directly if it has the requested data, or it can query the next DSA (recursively, as with DNS) in the chain to continue the search. The DoD requirements for the directory services included the support for both classified and unclassified data, as well as users with different clearances. [20]

Because X.500 was not designed to handle multiple classifications of data, the research is very applicable to the multilevel DNS problem. One analysis concluded that the most useful approach was to deploy high assurance servers on an MLS network where both high and low level data would be stored, and to have the DSA implementations run mostly as trusted subjects on the MLS servers, where the applications were aware of and helped to enforce the security policy [21]. The requesting clients would be single-level. However, recognizing that such high assurance MLS systems and software did not exist, the recommended short-term approach was to use high assurance MLS servers, each attached to several single-level networks, where MLS-ignorant single-level subjects (per access class) on the server could service the requests made by clients within the single-level networks. [21] These are both viable solutions for multilevel DNS under certain conditions.

5 Future Work

This research focused on environments where IP addresses and names are distinct across access classes, as one would expect from the IP specification. No analysis was performed with respect to polyinstantiation of addresses.

There cannot be a complete discussion of DNS without also considering how it affects the Dynamic Host Configuration Protocol (DHCP) [22]. This research should be expanded to consider the effects of the proposed solutions on DHCP. Is a MLS DHCP server necessary and, if so, how should it be designed to work with MLS DNS?

In addition to the impact on the DHCP protocol, additional research is needed to consider how MLS affects an implementation or configuration of the DNS Security Extensions (DNSSEC) [23]. The DNSSEC protocol uses public key technology to provide integrity and authenticity of DNS data.

An investigation to determine the characteristics of the current open source options for DNS daemons should be conducted. Which software is most conducive to being modified, e.g., which has the best documentation, best layering, the right combination of features, etc?

DNS has a little-used record type, LOC, which with appropriate authentication could be used to document the location of a particular network node, requiring longitude, latitude and altitude in its syntax. It may be possible to use this in some way to provide context-sensitive information for the GIG Vision Risk-Adaptive Access Control (RAdAC), which, among other things, uses the location of a subject to make access control decisions.

Research is needed to determine whether the current DNS protocol provides optional fields in the defined TCP/UDP packets that might be used in some way to communicate the access class of a DNS request with high assurance rather than basing the access class on the requestor’s address. In other words, is it possible to securely label a DNS request in some fashion without modifying the DNS header specification? At the same time, research can be performed on the IP header to determine if it can be used to communicate access class information securely. Encryption may be yet another option for communicating an access class based on a given key, or some other feature of encryption protocols, such as IPSEC.

[THIS PAGE IS INTENTIONALLY BLANK]

References

[1] Socolofsky, T., Kale, C., *TCP/IP Tutorial*, Request for Comments 1180, January 1991.

[2] Albitz, Paul, Liu, Cricket, *DNS and BIND*, O'Reilly and Associates, Inc., Sebastopol, CA, 2004.

[3] Internet Systems Consortium: A Brief History of BIND, http://www.isc.org/index.pl?/sw/bind

[4] *DNS for Rocket Scientists*, Appendix C: DNS Resources, http://www.zytrax.com/books/dns/apc/

[5] Deering, S., Hinden, R., *Internet Protocol, Version 6 (IPv6) Specification*, Request for Comments 2460, December 1998.

[6] *Secure64 DNA Authority*, http://www.secure64.com, datasheet S64DNS_2.4_v1.

[7] Common Criteria for Information Technology Security Evaluation, Version 2.3, August 2005, CCMB-2005-08-001.

[8] Lampson, B., *A Note on the Confinement Problem*, Communications of the ACM, Volume 16, Issue 10, October 1973, pp. 613-615.

[9] Bell, D.E., La Padula, L.J., *Secure Computer System: Unified Exposition and Multics Interpretation*, Electronic Systems Division, Air Force Systems Command, United States Air Force, Hanscom Air Force Base, Report MTR-2997 Rev. 1, march 1976.

[10] Irvine, C.E., *A Multilevel File System for High Assurance*, Proceedings of the 1995 IEEE Symposium on Security and Privacy, May 1995, pg. 78.

[11] Confirmed via e-mail communications with both a Navy security officer and an Air Force security officer.

[12] Nguyen, Thuy D., Levin, Timothy E., *MYSEA Testbed*, Proceedings from the 6th IEEE Systems, Man and Cybernetics Information Assurance Workshop, West Point, NY, June 2005, pp. 438-439.

[13] National Security Agency Information Assurance Directorate, *Executive Summary of the End-to-End IA Component of the GIG Integrated Architecture*, Version 1.0, October 26, 2004.

[14] *U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness*, National Information Assurance Partnership, version 1.03, June 29, 2007.

[15] Levin, Timothy E., Irvine, Cynthia E., Nguyen, Thuy D., *Least Privilege in Separation Kernels*, Proceedings of the International Conference on Security and Cryptography, Setubal, Portugal, August 2006, pp. 355-362.

[16] *Multilevel Data Management Security*, Committee on Multilevel Management Security, Air Force Studies Board, National Research Council, Washington, DC, 1983.

[17] Hinke T.H., Schaefer, M., *Secure Database Management System*, RADC-TR-75-266, Final Technical Report, System Development Corporation, November 1975.

[18] Denning, Dorothy, et. al., *A Multilevel Relational Data Model*, Proceedings of the 1987 IEEE Symposium on Security and Privacy, June 1987, pp. 220-234.

[19] Lunt, Teresa F., Schell, Roger R., Shockley, William R., *A Near-Term Design for the SeaView Multilevel Database System*, IEEE Symposium on Security and Privacy, April 1988, pp 234-244.

[20] Gaon, David, *Concept for Implementing a Globally Distributed X.500-Based DoD Directory*, Conference Record of the 1992 IEEE Military Communications Conference, Volume 3, pp. 1195-1199.

[21] Boucher, P.K., Lunt, T.F., *A Prototype Multilevel-Secure DoD Directory*, Proceedings of the 10th Annual Computer Security Applications Conference, December 1994, pp. 180-188.

[22] Droms, R., *Dynamic Host Configuration Protocol*, Request for Comments 2131, March 1997.

[23] Arends, R., Austein, R., Larson, M., Massey, D., Rose, S., *DNS Security Introduction and Requirements*, Request for Comments 4033, March 2005.

[24] Domain Name System, Security Technical Implementation Guide, Version 3, Release 0.1 (Final Draft), 2006-05-15, Defense Information Systems Agency.

Appendix A: Other Possible MLS DNS Architectures

This Appendix documents several alternative MLS DNS approaches that were considered when examining the needs of the MYSEA Testbed and the GIG Vision. Each approach is described, along with its advantages and disadvantages.

Approach 1: One COTS DNS Daemon Using DNS Views

In this approach there is one Commercial off-the-shelf (COTS) DNS daemon on each DNS platform and a data file for each access class. The access class of the DNS request is determined by the source IP address of the DNS request.

This approach can be implemented using a mechanism introduced by BIND 9 called *views*. Views allow a DNS server to respond differently to a DNS request, depending on the IP address of the requestor, as shown in Figure 4. The DNS data files are populated based on the access class of the IP address.

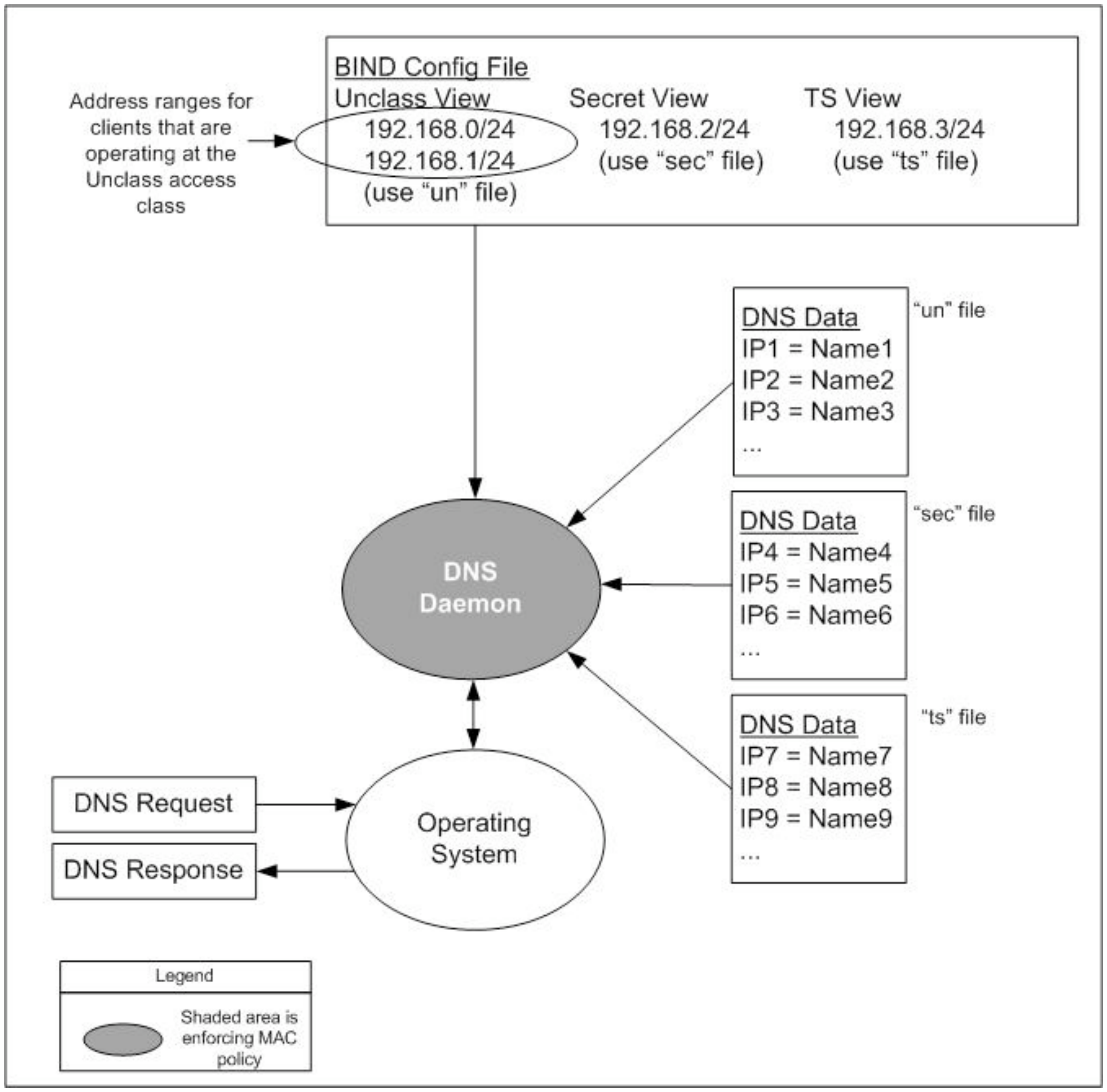

Figure 4. Approach 1: One COTS DNS Daemon Using DNS Views

The DNS daemon bases its view decision on the **source** IP address of the requestor, which indicates which DNS data file to refer to when searching for a match. This approach does not require the DNS daemon to directly know the access class of a DNS request; rather, that information is encoded in the source IP address and DNS configuration file.

Approach 1 Disadvantages

Views are established by specifying a range of IP addresses, such as "192.168.1/24" or by one of the following special tokens: "any", "none", "localhost", and "localnets". Therefore, the use of views will only work if the network address space is divided by access class, as currently done in the GIG. The number of supported access classes is therefore limited to defined ranges of addresses. In addition, a statically addressed client could not operate at multiple session levels, but instead must be tied to a specific access class. Therefore, an MSL client is only possible if the client OS uses a source IP address for network communications that corresponds to the access class of the requesting subject (i.e., a different IP address per access class).

This approach extends the security perimeter all the way to the application layer, requiring the potentially untrustworthy DNS application to make access control decisions. In other words, it is trusted to do the right thing but if not specially built and evaluated it will not be worthy of that trust. For example, there is no guarantee that the code is un-modifiable (e.g., free of buffer overflow vulnerabilities) and no guarantee that policy enforcement could not be bypassed. There is also no guarantee that there is no unspecified behavior in the code, such as Trojan horses or backdoors. If a high assurance version of DNS daemons was available to run on a high assurance OS, all these concerns become irrelevant. Unfortunately, such daemons do not currently exist.

Because there are no access class comparisons to determine whether the mapping should be provided to the requestor, the DNS daemon does not enforce MAC policy, but rather enforces a configuration. Therefore, though this approach appears to work, it is not a true MLS solution. Without modification of the BIND software, read-down is only possible if the information in lower-level views is copied to higher-level views. This problem could be minimized by having a script that merges the DNS mapping data together at DNS server startup. However, updates to the lower-level DNS data files made after daemon startup would not take effect at higher levels without an occasional resynchronization of the data files

Because the policy is being enforced based on the source IP address of the requestor, the underlying network access control would have to be able to detect a system that has had its IP address changed to an improper range for the device, and then disallow network communications. All other known types of IP spoofing attacks would also need to be addressed as well, to ensure the integrity of source addresses.

Though this approach initially appears to require no modifications to existing DNS software, this is false. When the daemon needs to perform a recursive DNS query there needs to be some mechanism for passing access class information (which in this case is the source IP address of the originating query) to the downstream DNS servers. Such a mechanism would require some level of modification to the existing DNS software. Without a high assurance DNS implementation or modification, this approach does not meet the high assurance requirement as stated in 2.3.

If the underlying system does not provide MLS functionality, then the OS cannot separate DNS data files based on the classification of the data. In such a situation the DNS daemon must be the only process running on the server. This will protect the DNS data files from other untrustworthy programs.

Approach 1 Advantages

With this approach, system administrators do not need to be concerned with the access class of DNS servers when configuring DNS clients. This removes the possibility of mis-configuring a client with the IP address of a higher-level DNS server.

No MLS servers are required to run the DNS daemon, though the enforcement method for a network access control policy may have its own MLS requirements.

Approach 2: One COTS DNS Daemon, Multiple Single-Level DNS Data Files

In this approach the DNS *view* feature is modified to use the access class of the requesting subject (rather than the source IP address) to determine the DNS data file to search. This approach, shown in Figure 5, is very similar to Approach 1 where the DNS daemon enforces the MAC policy.

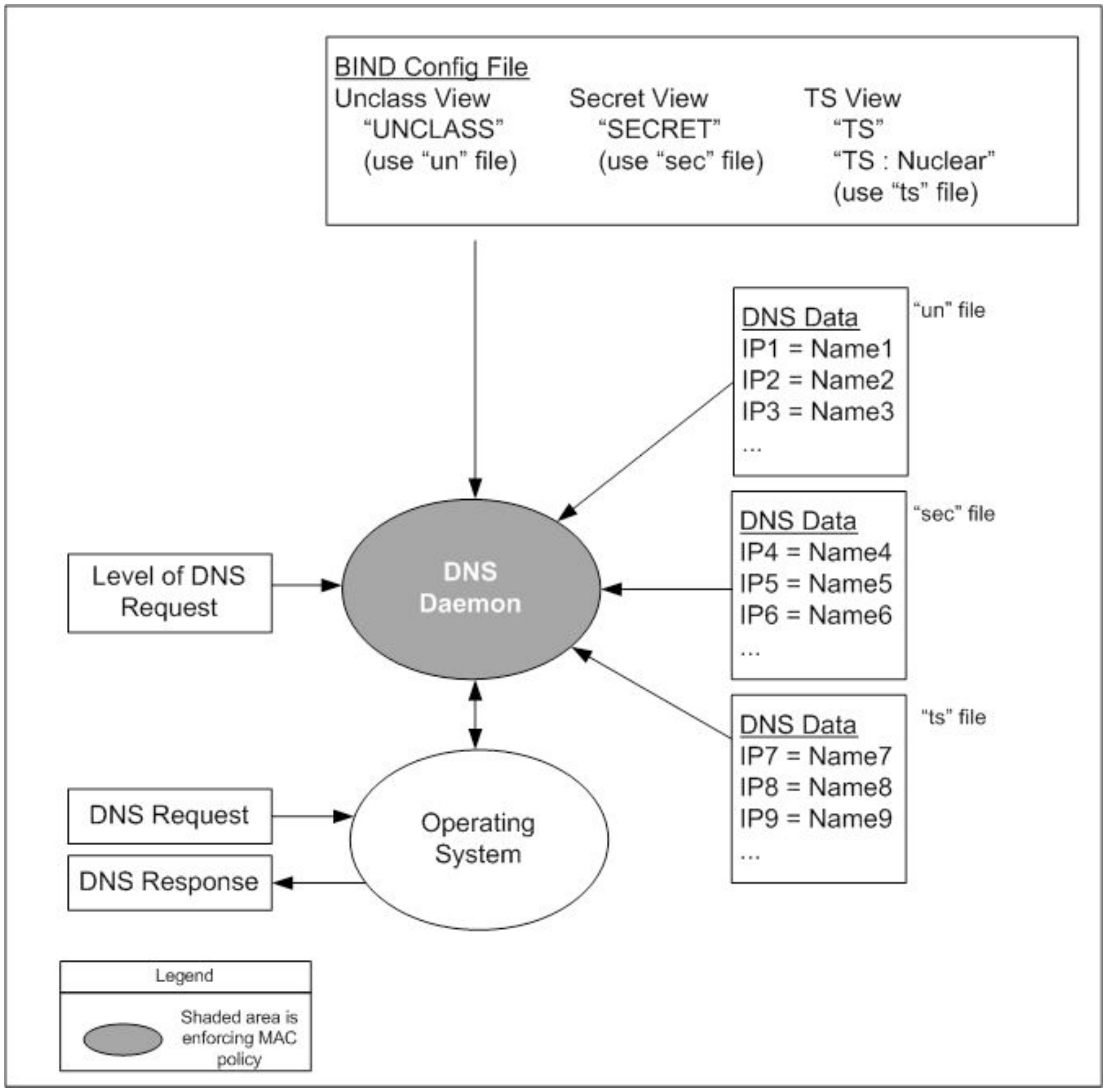

Figure 5. Approach 2: One COTS DNS Daemon, Multiple Single-Level DNS Data Files

Approach 2 Disadvantages

Since policy enforcement is being performed by an untrusted application, this approach has the same concerns as those described in Approach 1.

The existing DNS software would need to be modified to implement Approach 2. Even given that the code changes could be implemented successfully, the cost of maintaining and upgrading the software would have to be considered. These include fixing security-relevant bugs that have been found in BIND, or software updates as changes to the DNS protocol are ratified and new versions of BIND are released.

Without additional modifications to the DNS software, the daemon would not be able to resolve names that exist at a lower level, because the lower level DNS data are in different files. However, it may be possible to merge DNS data files appropriately at daemon startup. Changes made after startup would not take effect at higher levels without an occasional resynchronization of the data files.

Even though this approach does not require special MLS functionality from the underlying system, it should require a high level of assurance, since access control decisions are being made on the system. If a high assurance operating system is used, hardware choices will be limited because such operating systems are evaluated against a limited set of hardware. If the underlying system does not provide MLS functionality, then the configuration of the server should be restricted as described in Approach 1 Disadvantages. Without a high assurance DNS implementation or modification, this approach does not meet the high assurance requirement, as stated in 2.3.

In addition to DNS source code modifications this approach requires the development of trusted protocols and software to communicate the access class of the requester of the DNS translation to the DNS daemon and then to DNS servers that are queried to handle iterative requests.

Approach 2 Advantages

The hardware requirements for the DNS server in this approach would not be any greater than a “normal” DNS server because only one DNS daemon is needed per system.

With this approach, system administrators do not need to be concerned with the access class of DNS servers when configuring DNS clients. This removes the possibility of mis-configuring a client with the IP address of a higher-level DNS server.

The syntax of the DNS data files would not need to change, which makes the modification of the BIND source code easier than it otherwise would have been.

Approach 3: One DNS Daemon, One DNS Database

This approach uses a single modified COTS DNS daemon with an MLS DNS database. It uses one DNS daemon as a trusted subject to enforce the MAC policy with respect to name resolution. The daemon compares the access class of the DNS request to the access class of the stored name-to-address mapping to determine whether the information should be released to the requestor, as shown in Figure 6.

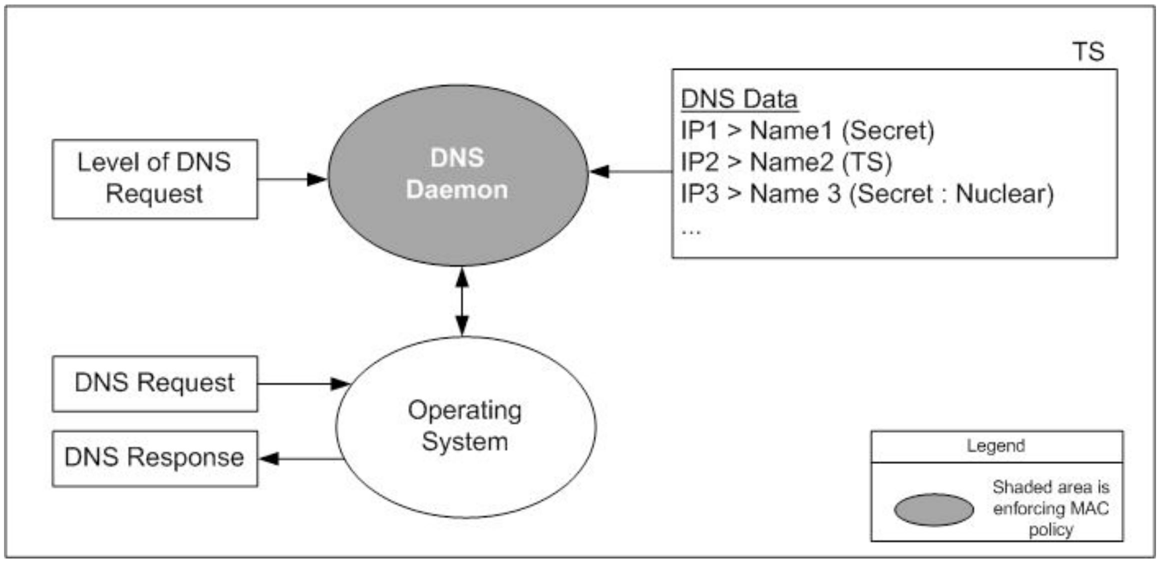

Figure 6. Approach 3: One DNS Daemon, One DNS Database

Approach 3 Disadvantages

This approach extends the security perimeter all the way to the application layer, which has the same concerns as those described in Approach 1 Disadvantages.

The existing DNS software would need to be modified to read a new set of configuration data to help enforce the new policy. In addition, changes would have to be made to support the passing of access class information when iterative DNS requests are sent to other DNS servers. Even if the code changes could be implemented successfully, the cost of maintaining and upgrading the software would have to be considered. These include fixing security-relevant bugs that have been found in BIND, or software updates as changes to the DNS protocol are ratified and new versions of BIND are released.

Even though this approach does not require special MLS functionality of the underlying system, it requires a high level of assurance, since access control decisions are being made on the system. If a high assurance operating system is used, hardware choices will be limited because such operating systems are evaluated against a limited set of hardware. If the underlying system does not provide MLS functionality, then the configuration of the server should be restricted as described in Approach 1 Disadvantages.

This approach also requires the development of trusted protocols and software to communicate the access class of the requester of the DNS translation to the DNS daemon and then to DNS servers that are queried to handle iterative requests.

Approach 3 Advantages

The hardware requirements for the DNS server in this approach would not be any greater than a “normal” DNS server because only one DNS daemon is needed per DNS system.

All DNS clients could be configured with the same single IP address for DNS resolution, without regard to the potential access class of the DNS request. This removes the possibility of mis-configuring a client with the IP address of a higher-level DNS server.

Approach 4: Many DNS Daemons on Many Physical Servers

In this approach there exists one DNS platform per access class, each with one DNS data file. DNS clients are configured to resolve queries with a server of the same access class, as shown in Figure 7.

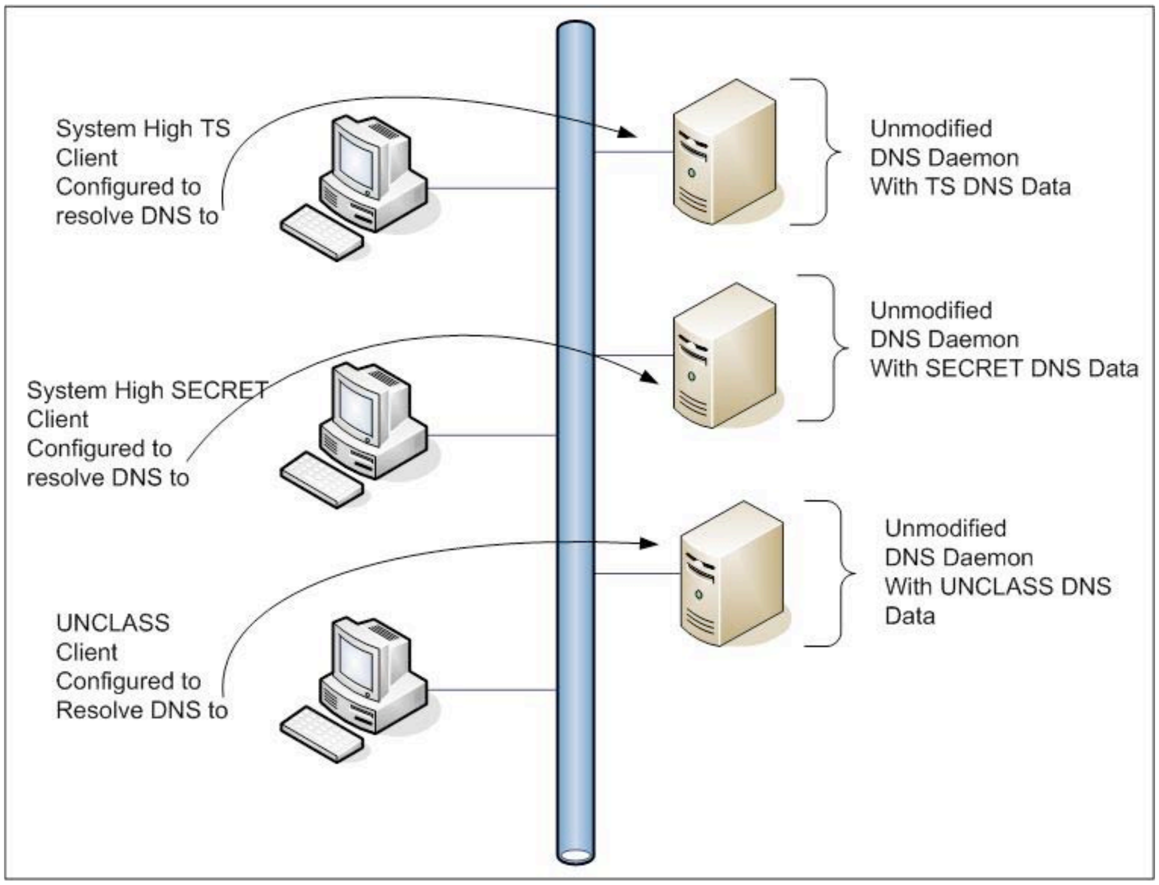
System High TS Client Configured to resolve DNS to
Unmodified DNS Daemon With TS DNS Data
Unmodified DNS Daemon With SECRET DNS Data
System High SECRET Client Configured to resolve DNS to
Unmodified DNS Daemon With UNCLASS DNS Data
UNCLASS Client Configured to Resolve DNS to

Figure 7. Approach 4: Many DNS Daemons on Many Physical Servers

Approach 4 Disadvantages

This is not a true MLS solution, but rather achieves separation of data by physical separation and configuration of components.

Supporting DNS requests from an MLS client becomes an issue because the destination IP address for the DNS server depends on a user's session level, which changes. An MLS client can only be supported in this approach if the client OS can dynamically change the destination DNS server IP address based on the session level of the requesting subject.

Supporting multiple physical DNS servers increases the cost of hardware, operation and maintenance. From a DNS point of view, there would be no cost savings by merging the currently separated DoD networks because there would still be (at least) as many DNS servers after the merge.

Each client system needs to be configured with the IP address of its local DNS server. In this approach, instead of that address being the same for all clients, it would change, depending on the highest-level of data the client is authorized to process and store, which introduces the potential for misconfiguration and leakage of data from high to low.

The DNS daemons supporting higher-level access classes would not be able to resolve names that exist at a lower level, because the lower level DNS data are on different machines.

Some method of identifying and handling inappropriate DNS requests must be developed, (e.g., a deliberately misconfigured lower-level client that tries to resolve names with a DNS server that is meant to service higher-level requests).

Approach 4 Variations

One variation to provide read down would be to manually merge lower-level information with the data maintained on the higher-level DNS servers, which is error prone, slow and costly.

Another variation is to have a high assurance administrative console that actually manages the DNS data files (i.e., a “hidden master”) and pushes updates out to the DNS servers, including lower-level data merged with the higher-level DNS data files.

Approach 4 Advantages

Absolutely no modifications to DNS software or file syntax will be required, greatly reducing the cost of implementation (to be balanced with the increased hardware costs), as well as the risk of implementation failure.

No expensive MLS servers are required to run the DNS daemon, though the method for enforcing a network access control policy may have its own MLS requirements. Otherwise, commodity hardware and operating systems can be used with this approach.

This approach can easily deal with the problem of DNS servers needing to perform iterative DNS queries; all servers are only configured to query DNS servers that store the same level of information. Therefore, no changes in DNS protocols are required to support it.

Approach 5: Treat all IP Addresses as Unclassified Data

Within a distributed MLS environment (e.g., enclave or group of enclaves) in which network addresses are not sensitive, a standard COTS DNS platform, daemons and data files can be used. If all DNS data is labeled Unclassified, then a DNS daemon could execute at the Unclassified level on an MLS server. A problem arises because subjects executing at a higher access class would be unable to query a DNS subject running at the lower level, because it would involve a write down, and would violate basic Bell-LaPadula rules. One solution is to execute the DNS daemon as a trusted subject that is MLS ignorant. It would be able to read the DNS data at all access classes as well as communicate with subjects at all access classes. An easier approach is to use a commodity OS that does not support MLS, and communicates with no regard to access class, but this introduces possibilities for information leakage channels.

If a network device was *named* in a way that revealed its access class or importance (e.g., veryimportantserver.navy.mil), then that is a problem of a different sort, and could be dealt with via policy statements, such as those found in DoD Operational Security (OPSEC) policies [24] that require network devices to be named in such a way that the names do not provide sensitive information about the devices.

Approach 5 Disadvantages

This approach would only be acceptable in environments in which IP addresses are not sensitive. In addition, in order to provide high assurance, the DNS daemon or the DNS resolver in the clients must be trustworthy, because the DNS daemon can communicate with subjects at all levels, introducing a channel for information leakage. This appears to remove what appeared to be the major advantage to this approach: use of COTS hardware and software to support DNS, because no high assurance DNS implementations exist, and these would therefore have to be implemented from scratch using high assurance methods.

Approach 5 Advantages

This approach will work in harmony with any other approach, even in the GIG, where various organizations may have different policies about IP address sensitivity, assuming each organization has control over its DNS servers.

This approach does not require any modifications to the DNS configuration and data files, but the need to re-write the DNS software makes the advantage somewhat irrelevant.

This approach does not need MLS functionality from the underlying OS, reducing the potential cost of the DNS servers. Commodity hardware and operating systems can be used with this approach. This approach is easy to administer, and does not require more resources than a “normal” DNS installation would require.

Approach 6: Development of a High Assurance DNS Daemon

Coverage of potential options would not be complete without considering the development of a trusted DNS daemon from scratch. In this approach there would be a single custom high assurance MLS daemon per DNS platform that manages a single MLS DNS database. There are two options: 1) develop an MLS-aware DNS daemon from scratch, but which depends on an underlying MLS Server to enforce the MAC policy (i.e., a high assurance variation of the recommended MYSEA solution, described in Section 3.1); or 2) develop a DNS daemon from scratch that runs as a trusted subject, enforcing its own policy (i.e., a high assurance version of Approach 3: One DNS Daemon, One DNS Database).

Approach 6 Disadvantages

Because of the size and complexity of the project, as explained in Section 1.4, it would be a relatively costly effort to both design and implement a replacement DNS daemon that supports MLS, in addition to the effort to do so in a high assurance manner.

If the underlying system does not provide MLS functionality, then the OS cannot separate DNS data files based on the classification of the data. In such a situation the DNS daemon should be the only process running on the server. This will protect the DNS data files from other untrustworthy programs.

This approach requires the development of a trusted way to communicate the access class of the DNS request to the DNS daemon, as described in Approach 2 Disadvantages.

Approach 6 Advantages

Developing a daemon using high assurance methods makes it possible to create software that can be trusted to enforce an MLS DNS policy.

By starting from scratch, the configuration file syntax could also be redesigned to allow for additional desired features, such as an access class designator. The existing DNS configuration syntax has a reputation for being complex, so a potential advantage to this approach would be DNS files that are easier to manage. Whether the existing configuration syntax is complex for a purpose, or complex because of backward compatibility, or a complex set of options, or whether its reputation for complexity is undeserved, would affect the difficulty of a syntax re-design.

Approach Comparison

Table 1 provides a quick comparison of the disadvantages of the approaches described in this paper. The approach properties are not equally weighted. Therefore, a “score” cannot be determined by just adding up the occurrences in each column. The legend for the table is given below.

1	One COTS DNS Daemon Using DNS Views
2	One COTS DNS Daemon, Multiple Single-Level DNS Data Files
3	One DNS Daemon, One DNS Database
4	Many DNS Daemons on Many Physical Servers
5	Treat all IP Addresses as Unclassified Data
6	Development of a High Assurance DNS Daemon
M	MYSEA Tested
G	DoD GIG Vision

Table 1. Approach Disadvantages

Properties	Approach							
	1	2	3	4	5	6[3]	M	G
DNS software modifications required	√	√	√					
Re-write of entire DNS application					√	√		
Trusted subject (enforcing policy)	√	√	√		√	√	√	
No run-time comparison of access classes	√	√		√	√			√
Approach is low assurance	√	√	√				√	
MLS OS required for server							√	√
Network access control required	√			√				√
Requires development of software and protocols outside of the DNS platform	√	√	√	√		√	√	
One DNS server IP address per access class				√				√

“Network access control required” refers to the necessity of having a mechanism that would prevent the spoofing of IP addresses for those approaches that rely on the source or destination IP address of a network packet to make access control decisions.

“Requires development of software and protocols outside of the DNS platform” refers to the need to develop software that is not directly tied to the modification or re-write of the DNS daemon code base.

[3] The properties of this approach are very dependent on the design choices made; this may not be a complete list of properties associated with Approach 6.

Table 2 restates the seven requirements set forth in Section 2.3 and identifies the approaches that meet each requirement. There are cases where the satisfaction of a requirement by an approach is subjective.

Table 2. Approaches mapped to Requirements

Requirements	Approach							
	1	2	3	4	5	6	M	G
1. Clients shall be able to resolve an address at any access class.	√	√	√	√	√	√	√	√
2. Iterative DNS queries shall preserve the access class of the initiating client.	√	√	√	√	√	√	√	√
3. DNS data shall be protected from observation and modification.	√	√	√	√	√	√	√	√
4. Clients can do a read-down of DNS data.	√	√	√	√	√	√	√	√
5. Access to DNS data is based on access class of 1) requester and 2) DNS data.	√	√	√	√	√	√	√	√
6. Assurance shall be applied appropriately.				√	√	√	√	√
7. Scalable.	√	√	√		√	√		√

Table 2 clearly shows that there are only three approaches that meet all the previously stated requirements: Approaches 5, 6, and the approach described for the GIG. However, the results of Table 2 must be balanced with the results of Table 1.

THIS PAGE IS INTENTIONALLY BLANK

Initial Distribution List

1.	Dudley Knox Library, Code 013 Naval Postgraduate School Monterey, CA 93943-5100	2
2.	Research Office, Code 09 Naval Postgraduate School Monterey, CA 93943-5138	1
3.	Paul C. Clark Code CS/Cp Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5118	1
4.	Dr. Cynthia E. Irvine Code CS/Ic Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5118	2
5.	David J. Shifflett Code CS Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5118	1
6.	Ed Bryant Unified Cross Domain Management Office Hyattsville, MD 20783	1
7.	Joseph DeHaven Director, NSA F81 Fort Meade, MD 20655	1
8.	Rob Dobry National Security Agency Fort Meade, MD 20655	1
9.	Neil Kittleson National Security Agency Fort Meade, MD 20655	1

10. Steve LaFountain — 1
    National Security Agency
    Fort Meade, MD 20655

11. John Mildner — 1
    SPAWAR
    Charleston, SC 29419

12. Dr. John Monastra — 1
    Aerospace Corporation
    Chantilly, VA 20151

13. Louanna Notragiacomo — 1
    The MITRE Corporation
    McLean, VA 22102

14. Dr. Ralph Wachter — 1
    Office of Naval Research
    Arlington, VA 22203